Server

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <stdio.h>

#include <unistd.h>

#define MAX\_MSG 100

#define SERVER\_ADDR "127.0.0.1"

#define SERVER\_PORT 1500

main ( ) {

int sd, newSd, cliLen, n;

struct sockaddr\_in cliAddr, servAddr;

char line[MAX\_MSG];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* build server address structure \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bzero((char \*)&servAddr, sizeof(servAddr));

servAddr.sin\_family = AF\_INET;

servAddr.sin\_addr.s\_addr = inet\_addr(SERVER\_ADDR);

servAddr.sin\_port = htons(SERVER\_PORT);

/\*

bzero((char \*)&servAddr, sizeof(servAddr));

servAddr.sin\_family = AF\_INET;

inet\_aton(SERVER\_ADDR, &servAddr.sin\_addr);

servAddr.sin\_port = htons(SERVER\_PORT);

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* create stream socket \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sd = socket(AF\_INET, SOCK\_STREAM, 0);

printf("successfully created stream socket \n");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* bind local port number \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bind(sd, (struct sockaddr \*) &servAddr, sizeof(servAddr));

printf("bound local port successfully\n");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* specify number of concurrent \*/

/\* clients to listen for \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

listen(sd,5);

while(1) {

printf("waiting for client connection on port TCP %u\n",SERVER\_PORT);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* wait for client connection\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

newSd = accept(sd, (struct sockaddr \*) &cliAddr, &cliLen);

if(fork()==0)

{

printf("received connection from host [IP %s ,TCP port %d]\n",

inet\_ntoa(cliAddr.sin\_addr), ntohs(cliAddr.sin\_port));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* wait for data from client \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

do{

memset(line,0x0,MAX\_MSG);

n=recv(newSd, line, MAX\_MSG, 0);

line[n]='\n';

printf("received from host [IP %s ,TCP port %d] : %s\n",

inet\_ntoa(cliAddr.sin\_addr), ntohs(cliAddr.sin\_port), line);

}while(abs(strcmp(line, "quit")));

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* close client connection\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

printf("closing connection with host [IP %s ,TCP port %d]\n",

inet\_ntoa(cliAddr.sin\_addr), ntohs(cliAddr.sin\_port));

close(newSd);

} }

}

Client

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netdb.h>

#include <stdio.h>

#include <unistd.h>

#define MAX\_MSG 100

#define SERVER\_ADDR "127.0.0.1"

#define CLIENT\_ADDR "127.0.0.1"

#define SERVER\_PORT 1500

#define CLIENT\_PORT 2500

int main()

{

int sd, rc, i;

struct sockaddr\_in clientAddr, servAddr;

char line[MAX\_MSG];

// build server address structure

bzero((char \*)&servAddr, sizeof(servAddr));

servAddr.sin\_family=AF\_INET;

servAddr.sin\_addr.s\_addr=inet\_addr(SERVER\_ADDR);

servAddr.sin\_port=htons(SERVER\_PORT);

bzero((char\*)&clientAddr, sizeof(clientAddr));

clientAddr.sin\_family = AF\_INET;

clientAddr.sin\_addr.s\_addr = INADDR\_ANY;

clientAddr.sin\_port = htons(0);

//create stream socket

sd = socket(AF\_INET, SOCK\_STREAM, 0);

printf("Successfully created stream socket \n");

// bind local port number

bind(sd, (struct sockaddr \*) &clientAddr, sizeof(clientAddr));

printf("Bound local port successfully\n");

connect(sd, (struct sockaddr \*) &servAddr, sizeof(servAddr));

printf("Connected to server successfully \n");

do

{

printf("Enter string to send to server : ");

scanf("%s", line);

send(sd, line, strlen(line) + 1, 0);

printf("Data sent (%s)\n", line);

} while(strcmp(line, "quit"));

printf("CLosing connection with the server\n");

close(sd);

return 0;

}

**OUtput**

